Binary Search Tree add — how much time?

Set of $n$ elements

1
3
7
15
31

relationship between $n$ and the depth $d$

$n = 2^d - 1$
(full binary tree)

Assume: bushy tree

time to insert $\approx$ depth $d$

Given $n$, what is $d$?

Defn. $\log_b n =$ power on $b$

$\log_2 8 = 3$ to get $n$

In our tree:

$d = \log_2 n$
What about remove?

log_2 n
What about intersection?
start from smallest # and work up

inorder traversal
visit(node) &
  visit(node.left);
  process node
visit(node.right);

preorder traversal
visit(node) &
  visit(node.left);
  process node
visit(node.right);

for intersection, check if the value is in the other list
for intersection, preorder would preserve structure
intersection: \( N \times \log_2 m \)

for each element of \( S \)
if \( T \) contains the element, put it in the result

for \( S \wedge T \)
\[ n = |S| \]
\[ m = |T| \]
Rep. E: Hash Table

hash function — computes a “random” number from the data value to be inserted or searched for.
deterministic:

$\text{hash}(x)$ returns the same value every time

Problem: collisions

$\text{hash}(w) \neq \text{hash}(x)$
Resolving collisions:

Idea 1: Go to the next empty slot
⇒ on searching, keep going through slots until either finding it or reaching an empty slot
⇒ have to deal w/ deleted items (treat them as non-empty for searching)

Idea 1b: Secondary hash function tells you where to go next
idea 2: Separate Chaining

```
| n | w | y | z |
```

Java, util, HashSet

new HashSet<String>

assume: hash table is big enough that the lists are very small (approaching 1)

add: ~1 step
contains: ~1 step
remove: ~1 step
intersection: \( n \times 1 \approx n \) steps

Fastest, but unordered
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{add}(x) \times &amp; S )</td>
<td>( \sim n )</td>
<td>( \sim n/2 )</td>
<td>( \sim n/2 + \log_2 n )</td>
<td>( \log_2 n )</td>
<td>1</td>
</tr>
<tr>
<td>( \text{contains}(x) \times &amp; S )</td>
<td>( \sim n )</td>
<td>( \sim n/2 )</td>
<td>( \sim \log_2 n )</td>
<td>( \log_2 n )</td>
<td>1</td>
</tr>
<tr>
<td>( \text{remove}(x) \times &amp; S )</td>
<td>( \sim n/2 )</td>
<td>( \sim n/2 )</td>
<td>( \sim n/2 + \log_2 n )</td>
<td>( \log_2 n )</td>
<td>1</td>
</tr>
<tr>
<td>( \text{intersect}(S) )</td>
<td>( S \cap T = \emptyset ), (</td>
<td>T</td>
<td>= m )</td>
<td>( \sim n \times m )</td>
<td>( \sim n + m )</td>
</tr>
</tbody>
</table>